

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平7-281596

(43) 公開日 平成7年(1995)10月27日

(51) Int.Cl.⁶

識別記号

庁内整理番号

F I

技術表示箇所

G 0 9 C 1/00

9364-5L

H 0 4 L 9/06

9/14

H 0 4 L 9/ 02

Z

審査請求 未請求 請求項の数13 O L (全 12 頁)

(21) 出願番号 特願平7-33219

(22) 出願日 平成7年(1995)2月22日

(31) 優先権主張番号 9 4 0 6 6 1 3 . 1

(32) 優先日 1994年4月5日

(33) 優先権主張国 イギリス (GB)

(71) 出願人 390009531

インターナショナル・ビジネス・マシーンズ・コーポレーション

INTERNATIONAL BUSINESS MACHINES CORPORATION

アメリカ合衆国10504、ニューヨーク州アーモンク (番地なし)

(72) 発明者 イアン・エドワード・ヨークスミス
イギリス エス053 5キユウエイ ハン
プシャー州チャンドラーズ・フォード ホ
コンブ・ロード 71

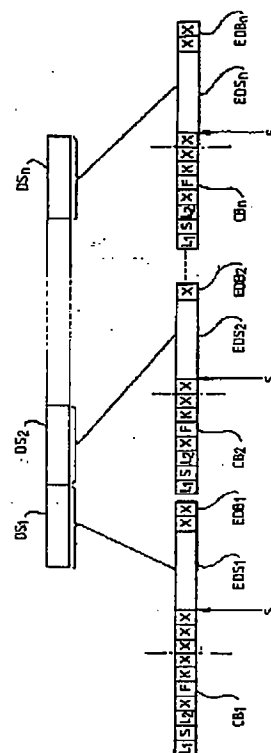
(74) 代理人 弁理士 合田 潔 (外2名)

(54) 【発明の名称】 暗号化方法およびシステム

(57) 【要約】

【目的】 本発明はデータを複数の制御データ・ブロックおよび暗号化データ・ブロックに暗号化する簡単な暗号化方法およびシステムを提供する。

【構成】 暗号化されるデータは可変長にすることができるデータ・セグメントに分割される。各制御ブロックはデータ・セグメントを暗号化するために使用される暗号化関数および関連するキー、暗号化データ・ブロック内の暗号化データ・セグメントの開始位置、ならびに暗号化データ・ブロックの長さなどの暗号化データ・ブロックに含まれているデータを解読するのに必要な情報を含んでいる。制御ブロックおよび暗号化データ・ブロックには両方とも、乱数が埋め込まれており、暗号化データ・ブロックを備えた暗号化データの開始位置は可変である。



【特許請求の範囲】

【請求項1】複数個のデータ・セグメント (DS_1 ないし DS_n) を含むデータを複数個のデータ・ブロック (EDB_1 ないし EDB_n) および関連する制御ブロック (CB_1 ないし CB_n) に暗号化する方法において、複数個の暗号化関数 (F_1 ないし F_j) の1つを選択し、選択した暗号化関数を使用してデータ・セグメントを暗号化して、暗号化データ・セグメントを形成し、暗号化データ・セグメントを含む暗号化データ・ブロックを作成し、暗号化データ・ブロックに関して、データを暗号化するのに使用された暗号化関数の指示を有する関連した制御ブロックを作成するステップからなる前記方法。

【請求項2】各データ・セグメントについての暗号化データ・ブロックの全長 (L_1)、各暗号化データ・ブロック内の各暗号化データ・セグメントの長さ (L_2) または暗号化データ・ブロック内の暗号化データ・セグメントの開始位置 (S) の少なくとも1つを選択するステップを含む、請求項1に記載の方法。

【請求項3】制御ブロックが、暗号化データ・ブロックの全長 L_1 、暗号化データ・セグメントの長さ L_2 または暗号化データ・ブロック内の暗号化データ・セグメントの開始位置 (S) の指示を含んでいる、請求項2に記載の方法。

【請求項4】暗号化データ・ブロックの全長 (L_1)、暗号化データ・セグメントの長さ (L_2)、または暗号化データ・ブロック内の暗号化データ・セグメントの開始位置 (S) がランダムに選択される、請求項2または3に記載の方法。

【請求項5】暗号化データ・セグメントを含んでいない暗号化データ・ブロックのフィールドを乱数 (X) で埋め込むステップを含む、請求項1ないし4のいずれか1項に記載の方法。

【請求項6】選択した暗号化関数とともに使用するために、複数個の暗号化キー (K_1 ないし K_j) から暗号化キーを選択するステップを含む、請求項1ないし5のいずれか1項に記載の方法。

【請求項7】制御ブロックが、前記選択した暗号化キー (K) の指示を含む、請求項6に記載の方法。

【請求項8】制御ブロックが、他の情報によって占められていないフィールドに乱数 (X) を含む、請求項1ないし7のいずれか1項に記載の方法。

【請求項9】複数個の所定の制御ブロック・フォーマット (CB_1 ないし CB_l) から1つを選択するステップをさらに含んでおり、各制御ブロックの所定の位置が制御ブロックの所定のフォーマットの指示 (C) を含む、請求項1ないし8のいずれか1項に記載の方法。

【請求項10】複数個のデータ・セグメント (DS_1 ないし DS_n) を含むデータを複数個の暗号化データ・ブロック (EDB_1 ないし EDB_n) および関連する制御ブ

ロック (CB_1 ないし CB_n) に暗号化するシステムにおいて、

各データ・セグメントに対して、複数個の暗号化関数

(F_1 ないし F_j) の1つを選択する手段と、

各データ・セグメントに対して、選択した暗号化関数を使用してデータ・セグメントを暗号化して、暗号化データ・セグメントを形成する手段と、

各データ・セグメントに対して、暗号化データ・セグメントを含む暗号化データ・ブロックを作成する手段と、

各データ・セグメントに対して、データを暗号化するのに使用された暗号化関数の指示を有する暗号化データ・ブロックに関連した制御ブロックを作成する手段とからなる前記システム。

【請求項11】複数個の暗号化関数 (F_1 ないし F_j) を使用して複数個のデータ・セグメント (DS_1 ないし DS_n) から、複数個の暗号化データ・ブロック (EDB_1 ないし EDB_n) および関連する制御ブロック (CB_1 ないし CB_n) に暗号化されたデータを解読する方法において、

制御ブロックおよび関連する暗号化データ・ブロックを読み取り、

関連する暗号化データ・ブロックとともに使用された制御ブロック内の情報から暗号化関数を決定し、

決定した暗号化関数に基づいて暗号化データ・ブロックからデータ・セグメントを解読するステップからなる前記方法。

【請求項12】制御ブロックが暗号化関数とともに使用する暗号化キーも含む、請求項11に記載の方法。

【請求項13】複数個の暗号化関数 (F_1 ないし F_j) を使用して複数個のデータ・セグメント (DS_1 ないし DS_n) から、複数個の暗号化データ・ブロック (EDB_1 ないし EDB_n) および関連する制御ブロック (CB_1 ないし CB_n) に暗号化されたデータを解読するシステムにおいて、

制御ブロックおよび関連する暗号化データ・ブロックを読み取る手段と、

関連する暗号化データ・ブロックとともに使用された制御ブロック内の情報から暗号化関数を決定する手段と、

決定した暗号化関数に基づいて暗号化データ・ブロックからデータ・セグメントを解読する手段とからなる前記システム。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明はデータ暗号化方法およびシステムに関する。

【0002】

【従来の技術】暗号化システムおよび方法は広い適用範囲を有している。たとえば、これらは交換されるデータが機密のものであり、データの機密性の品質を保存することが望ましいセルラ電話やローカル・エリア・ネット

ワークなどの通信システムに使用される。

【0003】暗号化システムの有効性は部分的には、用いられる暗号化方法の複雑度によって決定される。単純な先行技術の暗号化方法は、たとえば、英語のアルファベットの文字が置換アルファベットを形成することを含んでいる。暗号化対象のデータの各文字が、置換アルファベットから選択した対応する文字と置き換えられる。しかしながら、単純な暗号化方法は暗号化されたデータの無許可の受信者が簡単に復号しやすいものである。したがって、暗号化方法の複雑度は何年にもわたって高いものとなってきている。

【0004】暗号化方法の複雑度が高くなるにしたがい、データを暗号化し、その後復号するのにかかる時間も長くなる。短い暗号化時間を維持しながら、たとえば、暗号化方法を用いる通信システムの効率を高めるとともに、暗号化方法の高い複雑度を維持することが望ましい。暗号化時間の増加を補うために、しばしば、専用ハードウェアを使用して、暗号化復号化方法を実施することがある。米国特許第5257282号明細書は代数的に組み合わせられ、多重化されて、高速の複合コード・シーケンスを発生する複数個の低速シフト・レジスタからなる高速コード・シーケンス・コード・ジェネレータを開示している。特開平5-102960号明細書は、暗号規則が各通信の開始時にランダムに選択される暗号システムを開示している。同じ選択された暗号規則が通信全体にわたってデータを暗号化するために使用される。米国特許第5261003号明細書は、複数個のキーのうちの1個をデータのスクランブルに使用する、データのスクランブルによるデータ通信システムおよび方法を開示している。選択されるキーはスクランブル対象の入力データによって異なる。

【0005】受入れ可能な処理時間を達成するために専用ハードウェアで実現される、データ暗号化規格(DES)などのその他の暗号化方法も存在している。しかしながら、DESはセキュリティの制限を伴っており、その配布を制限するものとなっている。

【0006】

【発明が解決しようとする課題】したがって、従来技術には、無許可の受信者が解読するのが困難であり、暗号化および解読時間が比較的短い暗号化データをもたらす簡単な暗号化方法が欠けていた。

【0007】

【課題を解決するための手段】したがって、本発明は複数個のデータ・セグメントからなるデータを複数個の暗号化されたデータ・ブロック並びに関連する制御ブロックに暗号化するための方法であって、各データ・セグメントについて、複数個の暗号化関数の1つを選択し、選択した暗号化関数を使用してデータ・セグメントを暗号化して、暗号化データ・セグメントを形成し、暗号化データ・セグメントを含む暗号化データ・ブロックを作成

し、暗号化データ・ブロックに関して、データを暗号化するために使用された暗号化関数の指示を有する関連した制御ブロックを作成することからなる方法を提供する。

【0008】同一のデータ・セットに対して複数の暗号化技法を使用すると、使用する個々の暗号化技法が比較的単純であって、きわめて簡単かつ迅速に計算できるものであっても、無許可の解読がきわめて困難となる。このような暗号化プロセスは、たとえば、安全な通信や機密資料の格納などに広く適用できる。

【0009】データのセキュリティをさらに向上させるために、暗号化データ・ブロックの全長、暗号化データ・ブロック内の暗号化データ・セグメントの長さ、および暗号化データ・ブロック内の暗号化データ・セグメントの開始位置のうちの少なくとも1つの値を選択または変更できることが望ましい(本実施例において、これらのうち最初のものは固定されており、2番目および3番目が選択されるが、任意の組み合わせが選択可能である)。このような選択を所定のパターンまたは何らかの既知のパラメータ(たとえば、日付)にしたがって行うことができるが、好ましい解決策はランダムに生成された数(乱数)に基づいてこれらの値を選択することである(暗号化データ・セグメントの長さがこれを含んでいる暗号化データ・ブロックの長さよりも短いというような制約に、このような選択が合致していることが必要であることは明らかであろう)。

【0010】暗号化データの開始位置が暗号化データ・ブロック内で変化できる場合には、制御ブロックが、暗号化データ・ブロック内の暗号化データ・セグメントの開始位置の指示(indication)からなっていることが好ましい(ただし、この情報を何らかの独立した機構によって供給することができる)。同様に、制御ブロックは、必要に応じ、暗号化データ・ブロックの全長およびデータ・セグメントの長さの指示からなることもできる。すなわち、データ・セグメントの長さ、暗号化データ・ブロックの長さまたは暗号化データ・ブロック内の暗号化データ・セグメントの位置に可変値を使用した場合には、これらの値が制御ブロックに含まれていることが好ましい。

【0011】ほとんどの暗号化関数は暗号化キーを利用して、暗号化関数とともにデータを暗号化する。コードを解読するためには、暗号化関数とキーの両方が既知であって、これによってセキュリティを高めるものでなければならない。したがって、本発明はデータ・セグメントを暗号化するために使用される選択した暗号化関数とともに使用される複数個の暗号化キーから暗号化キーを選択するステップをさらに含めることができる。

【0012】暗号化データを解読するために暗号化データの供給者とその受領者との間で暗号化キーが交換される必要性をなくすために、制御ブロック内でデータを暗

号化するために使用される暗号化キーの指示を提供するのが望ましい。

【0013】固定フォーマットを有する制御ブロックは、暗号化される各データ・セグメントによってフォーマットが変化する制御ブロックよりも解読しやすい。

【0014】したがって、複数の所定の制御ブロック・フォーマットの1つを選択するステップをさらに含んでおり、各制御ブロックの所定位置が制御ブロックの所定のフォーマットの指示を含むようにすることもできる。

【0015】本発明は複数のデータ・セグメントからなるデータを複数の暗号化データブロックおよび関連する制御ブロックに暗号化するシステムにおいて、各データ・セグメントに対して、複数の暗号化関数の1つを選択する手段と、各データ・セグメントに対して、選択した暗号化関数を使用してデータ・セグメントを暗号化して、暗号化データ・セグメントを形成する手段と、各データ・セグメントに対して、暗号化データ・セグメントを含む暗号化データ・ブロックを作成する手段と、各データ・セグメントに対して、データを暗号化するために使用される暗号化関数の指示からなる暗号化データ・ブロックと関連した制御ブロックを作成する手段とからなるシステムも提供する。

【0016】本発明はさらに、複数の暗号化関数を使用して、複数のデータ・セグメントから複数の暗号化データ・ブロック並びに関連する制御ブロックに暗号化されたデータを解読するための方法において、制御ブロックと関連する暗号化データ・ブロックを読み取り、関連する暗号化データ・ブロックとともに使用される制御ブロック内の情報から暗号化関数を判定し、判定した暗号化関数に基づいて暗号化データ・ブロックからデータ・ブロックを解読するステップからなる方法を提供する。

【0017】

【実施例】図1には、暗号化されるデータ(D)が示されている。データは複数のデータ・セグメントに分割される。各データ・セグメント(DS)の長さは変動し、各データ・セグメントに対して生成されたそれぞれの乱数(L₂)によって判定される。暗号化データは各データ・セグメント(DS)に対する制御ブロック(CB)と暗号化データ・ブロック(EDB)からなっている。暗号化データ・ブロック(EDB)は暗号化データ・セグメント(EDS)を含んでいる(すなわち、元のデータ・セグメントを実際に含んでいるセグメントは暗号化されたフォーマットである)。制御ブロック(CB)は暗号化データ・ブロック(EDB)のデータ・バイトのフォーマットに関する情報、特にデータ・セグメント(DS)を暗号化するために使用される暗号化関数(F)と暗号化キー(K)、およびランダムに選択された暗号化データ・ブロック(EDB)内の暗号化データ・セグメント(EDS)の開始位置(S)の指示の情報

に関する複数のフィールドを含んでいる。制御ブロック(CB)および暗号化データ・ブロック(EDB)にも、乱数(X)が埋め込まれている。

【0018】暗号化データのフィールドは次の通りである。

L₁ = 暗号化データ・ブロック(EDB)の長さ

S = 暗号化データ・ブロック内の暗号化データ・セグメント(EDS)の開始位置

L₂ = 暗号化されるデータ・セグメント(DS)のバイト数

F = データ・セグメントの暗号化に使用される暗号化関数の指示

K = データ・セグメントの暗号化に使用される暗号化キーの指示

EDS = 暗号化データ・セグメント

X = 乱数

【0019】暗号化キーおよび関連する暗号化関数はデータ・セグメントの各バイトに対応する暗号化バイトに変換し、一般に、EDS=F(K, D)と表すことができる。ただし、EDSおよびKは上述の値を有しており、Dは暗号化されるデータ・セグメントである。EDSに対するDのマッピングは任意に選択することができる、バイトごとに行う必要はない。

【0020】適当な暗号化関数の例は次の通りである。

1. EDS=K 排他的OR D

2. EDS=Dを左へKビット、シフトする

3. EDS=D内のビット順を再整理する

【0021】図2を参照すると、利用可能な暗号化関数(F₁ないしF_j)および暗号化キー(K₁ないしK_j)の範囲から選択したさまざまな暗号化関数および暗号化キーが、各データ・セグメント(DS₁ないしDS_n)を暗号化するために使用されている。データ・セグメントを暗号化するために使用される暗号化関数は、所定の範囲(1ないしi)内の第1の乱数を生成し、この乱数を暗号化関数(F₁ないしF_j)の1つにマップすることによって決定される。同様に、データ・セグメントを暗号化するために使用される暗号化キーは、所定の範囲(1ないしj)から第2の乱数を選択し、この乱数を暗号化キー(K₁ないしK_j)の1つにマップすることによって決定される。したがって、各データ・セグメント(DS₁ないしDS_n)を暗号化した場合、全暗号化コードは、図3に示すように、複数の暗号化データ・ブロック(EDB₁ないしEDB_n)および関連する制御ブロック(CB₁ないしCB_n)からなる。

【0022】乱数と対応する暗号化関数または暗号化キーの間に1対1のマッピングは必要ない。特定の範囲の乱数を同一の暗号化関数または暗号化キーにマップし、これによって本発明の実施例を実現するのに必要な暗号化関数または暗号化キーの数を、後述の表1に示すように減少させることができる。

【0023】図4を参照すると、本発明による暗号化方法のステップを説明する流れ図が示されている。ステップ400において、データ・セグメントを暗号化するために使用される暗号化関数 (F_1 ないし F_j) を選択するための第1の乱数が所定の範囲から生成される。ステップ410は第2の所定範囲から第2の乱数を生成し、データ・セグメントを暗号化するために選択した暗号化関数によって使用される暗号化キー (K_1 ないし K_j) を選択する。暗号化データ・ブロック (EDB) の全長 (L_1) を決定するための所定の範囲内の第3の乱数 (L_1) がステップ420で生成される。ステップ430において、第3の乱数によって決定される範囲内の第4の乱数 (S) が生成され、これは暗号化データ・ブロック (EDB) における暗号化データ・セグメント (EDS) の開始位置を識別する。最後に、ステップ440は第3の乱数 (L_1) および第4の乱数 (S) によって決定される範囲内の第5の乱数 (L_2) を生成し、暗号化されるデータ・セグメントのサイズを決定する。

【0024】スペースが限定されている場合には、任意の適当な範囲を選択できるが、第4の乱数の選択範囲を $0 < S < L_1/2$ に、また第5の乱数の選択範囲を $(L_1 - S)/2 < L_2 < L_1 - S$ に制限するのが望ましい。

【0025】ステップ450は次の L_2 バイトのデータを取得して、データ・セグメントを形成する。データ・セグメントの各バイトはステップ460において、選択したそれぞれの暗号化関数と選択した暗号化キーを使用して暗号化される。暗号化データ・セグメントはステップ470において、対応する第4の乱数によって決定される開始位置から始まる暗号化データ・ブロックにおかれる。生成された乱数 (L_1 、 S 、 L_2 、 F および K) は次いで、所定のフォーマットにしたがって制御ブロックにおかれる。制御ブロックおよび暗号化データ・ブロックの他のフィールドには、図1に示すように、乱数 (X) が埋め込まれる。

【0026】本明細書記載の本発明の実施例は各暗号化データ・ブロックに対して同一フォーマットの制御ブロックを使用しているが、変動するフォーマットも同様に使用できる。このような場合には、各フォーマットは、図5に示すように、複数個の使用可能な制御ブロック・フォーマット (CD_1 ないし CD_1) の1つにマップされる所定の範囲から第6の乱数を生成することによって、複数個の制御ブロック・フォーマット (CB_1 ないし CB_1) から選択できる (L_1 、 S 、 L_2 、 F および K は上記と同じ意味を有する)。各制御ブロック (CB_1 ないし CB_1) は使用される制御ブロックの特定のフォーマットに識別を収めるための追加のフィールド (C) を必要とする。制御ブロックのフォーマットを変えるこのような技法を使用すると、制御ブロックの内容を解読する難度がさらに高くなり、それ故、暗号化データ・セグメントを解読する難度もさらに高くなる。あるいは、制御

ブロックのフォーマットの特定の順序が確立され、暗号化および解読両方の方法がその順序に適合するようになっている場合には、制御ブロックに含まれている情報が、利用されている特定の制御ブロックのフォーマットについての指示を与えるのを必要とせずに解読できる。

【0027】さらに、いくつかの乱数を生成する必要性および独立した制御ブロックとデータ・ブロックの必要性を未然に防ぐために、このような技法を使用することもできる。単一の乱数を使用して、暗号化データのフォーマット、上記の制御ブロックのフィールド、および組み合わされる暗号化データ・ブロックを識別することができる。各フィールドは暗号化ブロック全体のフォーマットを表す所定の値を含むこととなる。この技法による暗号化は対応する暗号化データ・フォーマットと関連する値をまず生成するという犠牲を払って、暗号化時間を短縮する。しかしながら、対応する暗号化データ・フォーマットおよび値は1回生成するだけですむ。

【0028】各暗号化データ (ED) をさらに処理するためレコードのファイルとして記憶媒体に書き込んだり、あるいはローカル・エリア・ネットワークその他の伝送媒体を使用して意図している受信者に伝送することができる。

【0029】図6を参照すると、暗号化データがレコードのファイルとして格納されているものとして、解読の流れ図が示されている。ステップ600において、暗号化データのレコードを含んでいるファイルがオープンされる。ステップ610はファイルから次のレコードを検索する。制御ブロックのフォーマットがわかっているもので、実施例が固定フォーマットの制御ブロックを有しているものとして、ステップ620は制御ブロックから値 L_1 、 S 、 L_2 、 F および K を抽出する。値 S および L_2 はステップ630で暗号化データ・セグメント (EDS) を識別するために使用される。 F および K の値は対応する暗号化関数の逆関数である解読関数をキー (K) とともに解読するために値 F をマッピングすることによって、ステップ640で暗号化データ・セグメント (EDS) を解読するために使用される。ステップ600ないし640はファイル内のすべてのレコードに対して反復される (L_1 の値を使用して、以降のデータ・セグメントに対する制御ブロックを見つけだせることに留意されたい)。

【0030】制御ブロックと暗号化データ・ブロックを同一のレコードと一緒に格納する代わりに、これらを個別に格納する本発明の実施例を実現することができる。各制御ブロックはレコードとして記憶され、暗号化データ・ブロックは連続したバイトとして別のファイルに格納される。したがって、復号データを制御ブロックから抽出したとき、次の L_1 バイトが暗号化データ・ブロックを含んでいるファイルから読み出される。 L_1 バイトは次いで上記のように解読される。

【0031】図7を参照すると、暗号化されるデータ(D)を含んでいるファイル(FF)を格納するためのディスク装置(DD)と、値 L_1 、S、 L_2 、FおよびKに対応した少なくとも5つの乱数を生成するための手段(RNG)と、暗号化されるデータ・セグメント内のバイト数に対応した次に利用可能な L_2 バイトを、ファイル(FF)から読み取るための手段(I)と、データを暗号化するために使用される暗号化関数を決定する手段(EF)と、データを暗号化するために使用される暗号化キーを決定する手段(EK)と、データ・セグメントを暗号化し、制御ブロック(CB)および暗号化データ・ブロック(EDB)からなる暗号化データを作成する

手段(E)と、暗号化データを含んでいるレコードを暗号化データ・ファイル(EDF)に書き込む手段(O)とからなる、本発明の実施例による暗号化システム(ES)が略示されている。

【0032】以下の表1は本発明の実施例を実施するために使用されるC言語のコードを示している。下記のC言語コードはC言語環境をサポートしているコンピュータで実行できる。適切なコンピュータは、たとえば、IBMパーソナル・コンピュータRS/2である。

【0033】

【表1】

```
(C) IBM Corporation 1994. All rights reserved.
/* レコード構造 */
typedef struct btrec      /* レコード */
{
    ULONG funcnum;        /* 関数番号0→なしにコード化する */
    ULONG key;            /* 関数に対するキー(no funcの場合、該当せず)
                           -最大キー値=64k-1 */
    ULONG length;         /* 次のデータの長さ */
    ULONG start           /* 暗号化データの開始位置 */
    ULONG cntlblock[20]   /* 制御ブロック */
    UCHAR data[500];      /* データ */
} BTREC;
typedef BTREC FAR * PBTREC;
/* 暗号化関数および解読関数 */
#define ENCODE1(d,k) (UCHAR) (d ^ (UCHAR)k)
#define ENCODE2(d,k) (UCHAR) ((UCHAR) (~d) ^ (UCHAR)k)
#define ENCODE3(d,k) (UCHAR) (d ^ (UCHAR)~k)
#define DECODE1(d,k) (UCHAR) (d ^ (UCHAR)k)
#define DECODE2(d,k) (UCHAR) (~(UCHAR) (d ^ (UCHAR)k))
#define DECODE3(d,k) (UCHAR) (d ^ (UCHAR)~k)
/* 名称: NewKey */
/* 記述: 変換に使用するキーを返す */
USHORT NewKey(VOID)
{
    USHORT key;          /* キー */
    /* 乱数からキーを解く */
    key=(USHORT)rand();
    /* 乱数を返す */
    return(key);
}
/* 名称: NewFunc */
/* 記述: 変換に使用する関数番号を返す */
ULONG NewFunc(VOID)
{
    ULONG func;          /* 関数番号 */
    /* 乱数から関数番号を解く */
    func=(ULONG)rand();
```

```

    if (func<10000) func=1;
    else if (func<20000) func=2;
    else func=3;
/* 関数番号を返す */
    return(func);
}
/* 名称:      NewLength */
/* 記述: 2つの境界の間のランダムな長さを返す */
/* Input:     下限 */
/*           上限 */
/* Returns:   Length */
SHORT NewLength(SHORT lower, SHORT upper)
{
    SHORT len;      /* 長さ */
    SHORT r;        /* 乱数 */
    SHORT i;        /* 作業変数 */
    /* Get a random number */
    r=(SHORT)rand();
    /* この範囲にマップし、長さを取得する */
    j=MAX_RANDOM_NUMER/(upper-lower+1);
    len=lower+r/j;
    return(len);
}
/* 名称:      FillStdData */
/* 記述: パターンを回避するための乱数を使用して、データ・レコードの標準
部品を埋め込み、暗号化関数と暗号化キーを選択する */
VOID FillStdData(PBTREC pbtrec)
{
    ULONG i;        /* Count */

/* レコードの標準部分に乱数を埋め込む */
    for(i=1; i<=500; i++)
        pbtrec->data[i]=(CHAR)rand();
    for(i=1; i<=20; i++)
        pbtrec->cntrlblock[i]=(ULONG)rand();
    pbtrec->start=0;
    pbtrec->length=NewLength(30, 100); /* 30-100の範囲の乱数を割り当てる */
    pbtrec->funcnum=NewFunc();
    pbtrec->key=NewKey();
    return;
}
/* 名称:      StartPos */
/* 記述: data[]内のデータの開始位置を決定する */
VOID StartPos(PBTREC pbtrec)
{
    /* データがアレイdata[]に合うように開始位置を選択する */
    while(pbtrec->start>500-pbtrec->length)
        pbtrec->start=(ULONG)rand();
    return;
}

```

```

/* 名称 :      ApplyFunc */
/* 記述 : 関数およびキーを適用することによってデータをコード化し、選択し
た開始位置でdata[]におく */
/* 入力 : レコードに対するポインタ */
VOID ApplyFunc(PBTREC pbtrec)
{
    ULONG i; /* Count */
/* 関数番号による処理 */
    if (pbtrec->funcnum==1)
    {
        for (i=0; i<pbtrec->length; i++)
            pbtrec->data[i+pbtrec->start]=ENCODE1(pbtrec->data[i],pbtrec->key)
    }
    else if (pbtrec->funcnum==2)
    {
        for (i=0; i<pbtrec->length; i++)
            pbtrec->data[i+pbtrec->start]=ENCODE2(pbtrec->data[i],pbtrec->key)
    }
    else if (pbtrec->funcnum==3 || pbtrec->funcnum != 0)
    {
        for (i=0; i<pbtrec->length; i++)
            pbtrec->data[i+pbtrec->start]=ENCODE3(pbtrec->data[i],pbtrec->key)
    }
    return;
}

/* 名称 :      TxData */
/* 記述 : 制御データを制御ブロックへ転送する */
VOID TxData((PBTREC pbtrec)
{
    pbtrec->cntrlblock[1]=pbtrec->length;
    pbtrec->cntrlblock[4]=pbtrec->start;
    pbtrec->cntrlblock[9]=pbtrec->funcnum;
    pbtrec->cntrlblock[15]=pbtrec->key;
    return;
}

```

【0034】レコード構造「btrec」は暗号化データを含んでいるレコードに含まれている情報のフォーマットを記述する。制御ブロックcntrlblock[]は、上記のF、K、L₂およびSに対応する後続のフィールド「funcnum」、「key」、「length」、「start」からの複数個の値からなっている。暗号化データ・ブロックはアレイ「data[500]」によって表され、したがって、L₁は500に固定されている（したがって、この値を制御ブロックに格納する必要はない）。上記したように、本発明は固定長の暗号化データ・ブロックに限定されるものではなく、可変長暗号化データ・ブロックを有する実施例を実現するように、上記コードは簡単に修正できる。

【0035】暗号化関数および解読関数は関数「ENCODE1」、「ENCODE2」、「ENCODE3」および「DECODE1」、「DECODE2」、「DECODE3」によって定義される。解読関数が暗号化関数の逆関数であることがわかる。ENCODE1はデータのバイトdと暗号化キーkの間のビット単位の排他的ORを実行する。ENCODE2はデータのバイトdの1の補数と暗号化キーkの間のビット単位の排他的ORを実行し、ENCODE3はデータのバイトdと暗号化キーkの1の補数の間のビット単位の排他的ORを実行する。解読関数「DECODE」は「ENCODE」関数の逆の演算を実行する。本発明は上記で画定した「ENCODE」および「DECODE」関数を使用することに限定されるものではない。

く、他の暗号化および解読関数を使用する実施例も実現できる。

【0036】関数「NewKey」は呼び出されるたびに新しい暗号化キーを生成する。本明細書記載の実施例において、キーは乱数である。

【0037】関数「NewFunc」はデータ・セグメントの暗号化にどの暗号化関数を使用されるかの指示を返す。3つの関数のうち1つが生成された乱数が属する範囲にしたがって選択される。

【0038】関数「NewLength」は「lower」および「upper」によって定義される所与の範囲内の乱数を返す。この関数はpbtrec→lengthを関数の戻り値に設定することによって、コード化されるデータ・セグメントの長さを決定するために使用される。本実施例において、長さが100という最大値に制限されるので、暗号化データ・ブロックのほとんどは実際には埋込みであることに留意されたい。スペースや帯域幅がさらに限定されている場合には、埋込み量は大幅に減らすことができる（さらには、省略される）。

【0039】関数「FillStdData」はアレイentrlblock[]およびdata[]を乱数で初期化し、開始位置データをゼロに設定し、データ・セグメントの長さを選択し、データ・セグメントを暗号化するのに使用される暗号化関数とキーを選択する。

【0040】関数「StartPos」は暗号化データ・セグメントが常にデータ・アレイdata[]に合致するようなものであるアレイdata[i]内の暗号化データ・セグメントに対するランダムな開始位置を生成する。

【0041】関数「ApplyFunc」は選択した暗号化関数およびキーをデータ・セグメント内の各バイトに適用することによってデータ・セグメントを暗号化する。選択した暗号化関数によれば、3つの「ENCODE」関数のうちの1つが呼び出され、暗号化されるデータdata[i]および暗号化キーpbtrec → keyをこの関数に渡す。暗号化データはレコードbtrecのデータ・アレイpbtrec → data[i + pbtre → start]に格納される。

【0042】関数「TxData」は制御情報をアレイ「entrlblock[]」に転送し、制御ブロックが固定フォーマットを有するようにする。次いで、制御ブロックおよび暗号化データ・ブロックを意図している受信者に伝送したり、あるいは後で適宜処理するために格納することができる。さらに、制御ブロックと暗号化データ・ブロックを個別に格納したり、伝送したりすることができる。

【0043】したがって、2つのアレイentrlblock[]およびdata[]は、解読を行うことのできる暗号化情報と暗号化データ・ブロックをそれぞれ含んでいる。

【0044】本明細書記載の本発明の実施例は一時に1つのバイトで作動する暗号化および解読関数を使用するが、暗号化および解読関数は一時にいくつかのバイトで作動することもできる。

【0045】さらに、複数の個別のデータ・セグメントを暗号化し、同一の暗号化データ・ブロックに格納できるが、各々が制御ブロックに格納されているそれぞれの暗号化関数、キー、開始位置および長さを有している。

【0046】まとめとして、本発明の構成に関して以下の事項を開示する。

【0047】(1) 複数のデータ・セグメント(DS₁ないしDS_n)を含むデータを複数のデータ・ブロック(EDB₁ないしEDB_n)および関連する制御ブロック(CB₁ないしCB_n)に暗号化する方法において、複数の暗号化関数(F₁ないしF_j)の1つを選択し、選択した暗号化関数を使用してデータ・セグメントを暗号化して、暗号化データ・セグメントを形成し、暗号化データ・セグメントを含む暗号化データ・ブロックを作成し、暗号化データ・ブロックに関して、データを暗号化するのに使用された暗号化関数の指示を有する関連した制御ブロックを作成するステップからなる前記方法。

(2) 各データ・セグメントについての暗号化データ・ブロックの全長(L₁)、各暗号化データ・ブロック内の各暗号化データ・セグメントの長さ(L₂)または暗号化データ・ブロック内の暗号化データ・セグメントの開始位置(S)の少なくとも1つを選択するステップを含む、上記(1)に記載の方法。

(3) 制御ブロックが、暗号化データ・ブロックの全長L₁、暗号化データ・セグメントの長さL₂または暗号化データ・ブロック内の暗号化データ・セグメントの開始位置(S)の指示を含んでいる、上記(2)に記載の方法。

(4) 暗号化データ・ブロックの全長(L₁)、暗号化データ・セグメントの長さ(L₂)、または暗号化データ・ブロック内の暗号化データ・セグメントの開始位置(S)がランダムに選択される、上記(2)または(3)に記載の方法。

(5) 暗号化データ・セグメントを含んでいない暗号化データ・ブロックのフィールドを乱数(X)で埋め込むステップを含む、上記(1)ないし(4)のいずれか1項に記載の方法。

(6) 選択した暗号化関数とともに使用するために、複数の暗号化キー(K₁ないしK_j)から暗号化キーを選択するステップを含む、上記(1)ないし(5)のいずれか1項に記載の方法。

(7) 制御ブロックが、前記選択した暗号化キー(K)の指示を含む、上記(6)に記載の方法。

(8) 制御ブロックが、他の情報によって占められていないフィールドに乱数(X)を含む、上記(1)ないし(7)のいずれか1項に記載の方法。

(9) 複数の所定の制御ブロック・フォーマット(CB₁ないしCB_j)から1つを選択するステップをさらに含んでおり、各制御ブロックの所定の位置が制御ブロッ

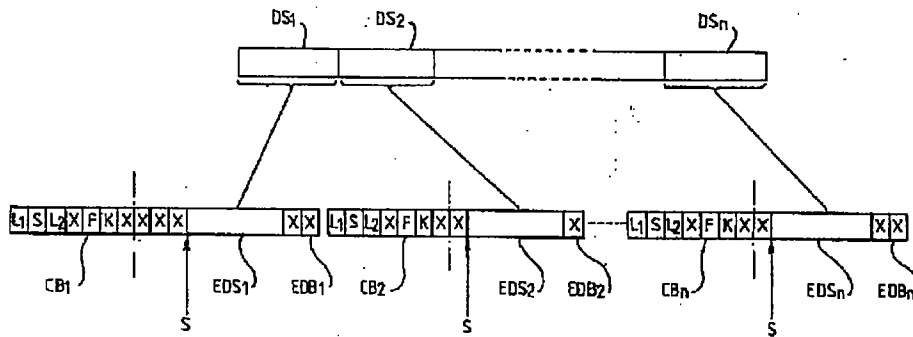
(13) 複数個の暗号化関数 (F_1 ないし F_i) を使用して複数個のデータ・セグメント (DS_1 ないし DS_n) か

【図 7】 暗号化システムの略図である。

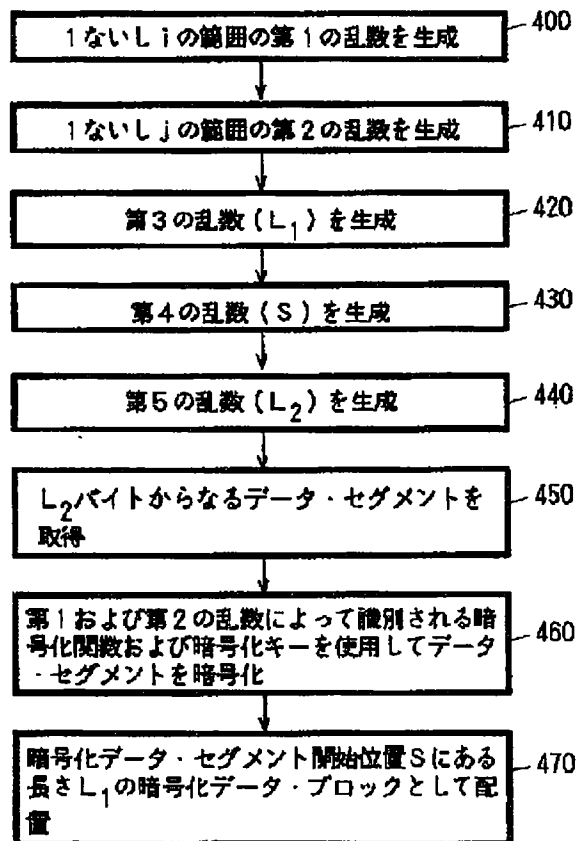
【图2】

直数 F	暗号化関数	直数 K	暗号キ一
1	F_1	1	K_1
2	F_2	2	K_2
⋮	⋮	⋮	⋮
i	F_i	j	K_j

【図3】



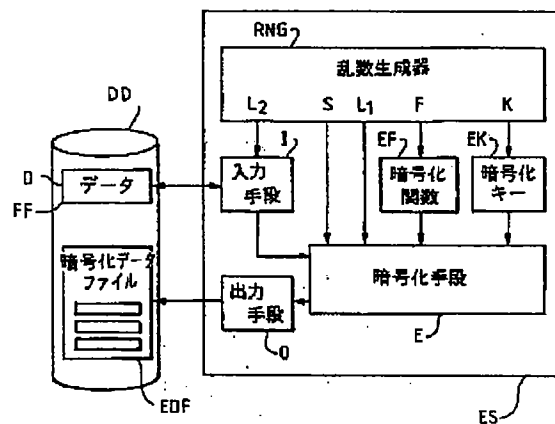
【図4】



【図5】

乱数	制御ブロック フォーマット
1	<div style="border: 1px solid black; padding: 2px; display: inline-block;"> L_1 S X L2 X F X K C </div> CB1
2	<div style="border: 1px solid black; padding: 2px; display: inline-block;"> X L1 X S L2 K X F C </div> CB2
...	...
i	<div style="border: 1px solid black; padding: 2px; display: inline-block;"> F X K S L1 X X L2 C </div> CBi

【図7】



【図6】

